PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 5/01, H04B 1/00 — A1

(11) International Publication Number: WO 98/53393

(43) International Publication Date: 26 November 1998 (26.11.98)

(21) International Application Number: PCT/US98/09033

(22) International Filing Date: 7 May 1998 (07.05.98)

(30) Priority Data:
08/862,648 23 May 1997 (23.05.97) US

(71) Applicant: ADOBE SYSTEMS INCORPORATED [US/US]; 345 Park Avenue, San Jose, CA 95110 (US).

(72) Inventor: RAMAN, T., V.; 345 Park Avenue, San Jose, CA 95110 (US).

(74) Agent: BOROVOY, Roger, S.; Fish & Richardson P.C., Suite 100, 2200 Sand Hill Road, Menlo Park, CA 94025 (US).

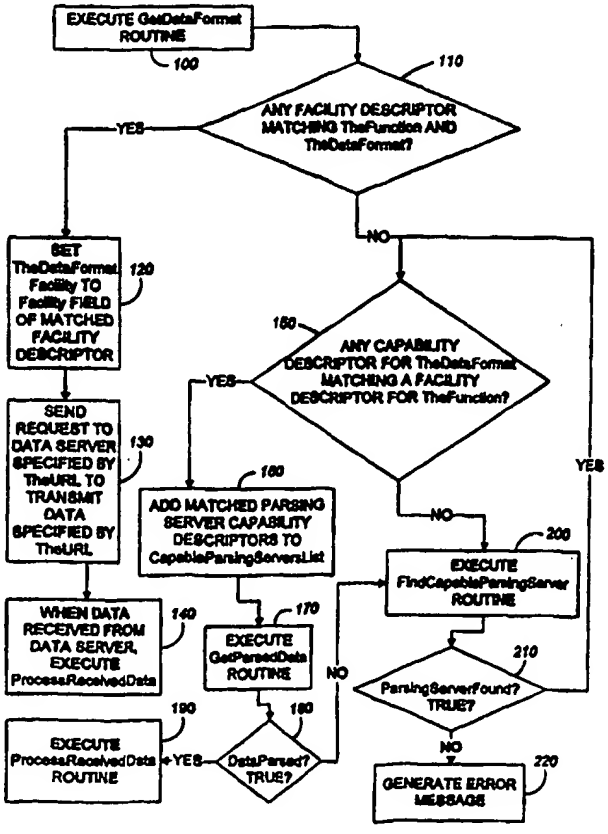
(81) Designated States: CA, JP, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: DATA STREAM PROCESSING ON NETWORKED COMPUTER SYSTEM LACKING FORMAT-SPECIFIC DATA PROCESSING RESOURCES

(57) Abstract

In a data network including server systems and client systems, a method of converting data from a first fromat unusable by a client system into a second format usable by the client system comprising the steps of sendig the data to a parsing server capable of converting such data from the first format to the second format, parsing the data in the parsing server from the first format to the second format, and sending the data in the second format to the client system.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

DATA STREAM PROCESSING ON NETWORKED COMPUTER SYSTEM LACKING FORMAT-SPECIFIC DATA PROCESSING RESOURCES

Background

The invention relates to processing data on a networked computer system lacking format-specific data processing resources.

Wide area networks (WANs), such as the Internet, facilitate the exchange of electronic data between heterogeneous computer systems. Through client/server mechanisms such as HyperText Transfer Protocol (HTTP) or File Transfer Protocol (FTP), data created or stored on one computer system can be made readily available for transfer to and use on another computer system.

In particular, computer systems on a WAN can be configured to operate as "clients" and "servers." With HTTP, for example, clients generate request messages which are sent over the WAN to servers. The request messages include a Uniform Resource Locator (URL), an alphanumeric string uniquely identifying the location of a data resource (*e.g.*, a data file or a process for generating a stream of data) on the WAN. When a request message is received by a server, the server fetches data from the requested resource and transmits the data to the client.

One impediment to such an exchange, however, is data format incompatibility. Modern software programs, such as word processors, desktop publishing systems and multimedia systems, represent and store data in a wide variety of formats. The format of data determines how it should be interpreted in order to make it intelligible to a human user. A user wishing to perform a function (*e.g.*, display, print, or edit) on data in a given format must therefore have access to facilities (*e.g.*, software, printers, or other hardware) that can process (*i.e.*,

properly interpret) data in that format. If the user's computer system does not include facilities having the ability to process data in a given format, the user will not be able to use the data.

Because of the multiplicity of data formats currently used in conjunction with commercial computer systems, standard conventions have been developed which permit efficient data format identification. One such convention involves the use of filename extensions to indicate the format of data within the file. For example, a file with the name of "picture.gif" has the filename extension "gif" indicating that the file contains data in GIF format. Another widely used convention is Multipurpose Internet Mail Extensions (MIME). Under the MIME framework, data formats are registered with a central authority which issues a set of standard codes for identifying the formats. MIME codes are often used in network communications to identify the format of data being transmitted between computer systems.

Within a single computer system or organization, the problem of data format incompatibility is likely to be relatively insignificant since it is likely that all data created on the system for a given purpose will be created using the same software program and will thus have the same format. However, on a WAN connecting heterogeneous computer systems, the inability of computers systems to process data formatted elsewhere will likely be more significant.

One conventional approach to the problem of data format incompatibility is to maintain, on a given computer system, facilities capable of recognizing all of the data formats likely to be encountered by users of that system. This approach is generally not practical, however, in light of the expense involved, the need for

users to learn how to operate multiple facilities, and the need to frequently add new facilities in light of the introduction of new data formats.

Another approach is to maintain on the given computer system a set of parser (data conversion) programs which are capable of converting data from one format to another. A user seeking to use data of a type which could not be processed by system's resident facilities would use a parser to translate the data into a format which could be. There are at least two disadvantages to this solution: First, the use of the parser will consume additional computer system resources since it must be executed every time data of a format which cannot be directly processed by the system is retrieved from the WAN. Moreover, the constantly changing variety and number of data formats makes the task of maintaining the ability to translate all necessary data formats a difficult one.

Summary of the Invention

In general, in one aspect, the invention features, in a data network in which a multiplicity of data formats exist, a method for performing a particular function on data represented in a first data format, on a client system lacking resources to perform the function on data in the first data format but having resources to perform the function on data in a second data format. The method includes transmitting a request message from the client system to a parsing server, transmitting the data from a data server to the parsing server, parsing the data at the parsing server into the second data format, receiving on the client system the data parsed into the second format from the parsing server, and using a resource on the client system to perform the function on the parsed data. Preferred embodiments of the invention include one

or more of the following features. The request message includes a Uniform Resource Locator specifying the location of the data on the network. The request message includes the data. The request message specifies the second data format. The request message specifies the first data format. The parsing server is selected from a pre-existing list of parsing servers. The parsing server is selected from among a set of parsing servers that respond to a request broadcast to the network. The pre-existing list of parsing servers is generated using information broadcast to the network by parsing servers. The pre-existing list of parsing servers is generated using information sent by parsing servers in response to messages broadcast to the network by the client system. The step of selecting the parsing server includes identifying the first data format, identifying which resources on the client can perform the function, identifying a set of data formats upon which the resources can perform the function, identifying the translation capabilities of parsing servers on the list, identifying a set of parsing servers that are capable of translating data from the first data format to one of the set of data formats and choosing a parsing server from the set. The step of identifying the translation capabilities of parsing servers includes accessing data stored on the client system. The step of identifying the translation capabilities of parsing servers includes sending messages to parsing servers over the network requesting them to indicate their translation capabilities and receiving messages from parsing servers indicating their capabilities. The network is a wide area network. The network is a local area network.

In general, in another aspect, the invention features a computer program, residing on a computer-readable medium, including instructions for causing a

computer connected to a network to receive a request message from a client system over the network, receive data from a data server in a first format over the network, parse the data into a second data format, and transmit the data in the second format over the network to the client system.

In general, in another aspect, the invention features a parsing server for use on a network, including a digital computer and a computer program storage medium having a computer program stored thereon for execution on the digital computer. The computer program includes instructions for causing the computer to receive a request message from a client system over the network, receive data from a data server in a first format over the network, parse the data into a second data format, and transmit the data in the second format over the network to the client system.

In general, in another aspect, the invention features, in a network in which a multiplicity of data formats are recognized, a method for performing a particular function on data represented in a first data format, on a client system lacking resources to perform the function on data in the first data format, but having resources to perform the function on data in at least one other data format. The method includes identifying the first data format, identifying which resources on the client can perform the function, identifying a set of data formats upon which the resources can perform the function, identifying the translation capabilities of parsing servers on a list of parsing servers, identifying the set of parsing servers which are capable of translating data from the first data format to one of the set of data formats, choosing a parsing server from among the set of parsing servers, sending a request message from the client system to the chosen parsing server,

which specifies the URL of the data, the first data format, and the second data format, sending the data specified by the URL from the data server to the parsing server, on the parsing server parsing the data into the second data format, sending the parsed data in the second data format from the parsing server to the client system, and using a resource on the client system to perform the function on the parsed data in the second data format.

Among the advantages of the invention are one or more of the following. Data can be readily exchanged among computer systems on a WAN notwithstanding the use of heterogeneous data formats on the WAN. The system load on a WAN for parsing data formats can be distributed among multiple servers. A computer system can perform functions on a multiplicity of data formats without maintaining on that system facilities capable of operating on all such formats. A computer system can perform functions on a multiplicity of data formats without using system resources to parse data into usable formats. Facilities for parsing a given data type need not be replicated on multiple systems on a WAN. Facilities for parsing a given data type on a WAN may be upgraded or modified without the need to upgrade or modify such facilities on all systems connected to the WAN.

Other features and advantages of the invention will become apparent from the following description and from the claims.

Brief Description of the Drawings

FIG. 1a is a schematic diagram of a WAN.

FIG. 1b is a flowchart showing the overall operation of the invention.

FIG. 2 is a schematic diagram of a client system.

FIG. 3 is a schematic diagram of a parsing server.

FIG. 4 is a schematic diagram of a data server.

FIG. 5 is a flowchart showing the steps taken by a client system in performing a given function on a specified resource.

FIG. 6 is a flowchart showing the steps taken by the routine GetDataFormat on a client system.

FIG. 7 is a flowchart showing the steps taken by the routine GetParsedData on a client system.

FIG. 7a is a schematic diagram of a PARSE_AND_TRANSMIT message.

FIG. 7b is a schematic diagram of a PARSE_AND_TRANSMIT_RESPONSE message.

FIG. 8 is a flowchart showing the steps taken by the routine ProcessReceivedData on a client system.

FIG. 9 is a flowchart showing the steps taken by the routine FindCapableParsingServer on a client system.

FIG. 9a is a schematic diagram of a PARSING_SERVER_CAPABILITY_QUERY message.

FIG. 9b is a schematic diagram of a PARSING_SERVER_CAPABILITY_RESPONSE message.

FIG. 10 is a flowchart showing the steps taken by the routine UpdateParsingServerList on a client system.

FIG. 11 is a flowchart showing the steps taken by a parsing server in response to a PARSE_AND_TRANSMIT message.

FIG. 12 is a flowchart showing the steps taken by a parsing server in response to a PARSING_SERVER_CAPABILITY_QUERY message.

Description of the Preferred Embodiments

Shown in FIG. 1a is a simplified example of a WAN 13, including client systems 14, data servers 15 and parsing servers 20. Note that any computer connected to the WAN may act as a client system, a data server or a parsing server, or any combination of them.

Shown in FIG. 1b is a flowchart showing the

overall operation of the invention. Data in a first format, which cannot be used on a client system, is transmitted to a parsing server capable of converting such data from the first format to a second format which can be used by the client system (step 16). The data is then parsed on the parsing server into the second format (step 17), and transmitted to the client system (step 18).

Shown in FIG. 2 is a simplified diagram of a client system 14. A client system 14 maintains data structures including TheFacilitiesList 25, TheParsingServerList 30, TheExtensionList 35, and a set of Facilities 40. TheFacilitiesList includes a list of Facility Descriptors 45. Each Facility Descriptor 45 includes a Function field, containing a Function_ID, a Format field, containing a Format_ID, and a Facility field, containing a Facility_ID. A Function_ID identifies a function which can be performed on data, such as printing, viewing, or scanning for text. A Format_ID corresponds to a data format. A Facility_ID identifies a facility located on the client system. Each Facility Descriptor indicates that the specified Facility is capable of performing the specified Function on data of the specified Format.

TheParsingServerList 30 includes a list of parsing server Capability descriptors 50. Each parsing server capability descriptor 50 includes a ConvertFrom field and a ConvertTo field, both containing a Format_ID. Each capability descriptor also includes a parsing server field, containing a Parsing_Server_ID. Each Parsing_Server_ID uniquely identifies a parsing server connected to the WAN. Each parsing server capability descriptor indicates that the specified parsing server is capable of converting data from format ConvertFrom to data of format ConvertTo.

TheExtensionList includes a list of Extension Equivalents 55. Each Extension Equivalent includes an Extension, containing an alphanumeric string, and a Format, containing a Format_ID. Each Extension Equivalent indicates that the presence of the given Extension in a URL means that the URL locates data in the specified Format.

Shown in FIG. 3 is a simplified diagram of a parsing server 20. A parsing server maintains data structures including TheParserList 60, and a set of Parsers 65. TheParserList includes a list of Parser capability descriptors 70. Each Parser capability descriptor 70 includes a ConvertFrom field and a ConvertTo field, both containing a Format_ID. Each Parser capability descriptor 70 also contains a Parser field containing a Parser_ID, which uniquely identifies a Parser located on a parsing server. Each Parser capability descriptor indicates that the specified Parser is capable of converting data from format ConvertFrom to format ConvertTo.

Shown in FIG. 4 is a simplified diagram of a data server 15. A data server maintains a set of Data Resources 70. Each Data Resource is either a data file, or a process which produces a sequence of data. Each Data Resource is associated with a URL 75, which uniquely identifies the Data Resource on the WAN, and a Format_ID 80.

Shown in FIG. 5 is a flowchart showing the steps taken by a client system in performing a given function (specified by the variable TheFunction), on a resource specified by a given URL, TheURL. (The client system may receive the URL from a variety of sources. For example, it may be specified by a user, or it may be specified by a hypertext link.) First, the routine GetDataFormat is called (step 100). As explained in more detail below,

GetDataFormat sets a global variable TheDataFormat equal to the Format_ID representing the data format of the data to be retrieved from TheURL. Next, TheFacilitiesList is checked to see if any of the Facility Descriptors has a Function field matching TheFunction, and a Format field matching TheDataFormat (step 110).

If a match is found, the global variable TheDataFormatFacility is set equal to the Facility field of the matched Facility Descriptor (step 120). Next, the client system sends a request to the data server specified by TheURL, requesting the data server to transmit the data specified by the URL (step 130). When the data is received, the routine ProcessReceivedData is executed (step 140).

If no match is found on TheFacilitiesList, TheParsingServerList is checked to see if any of the parsing server capability descriptors has a ConvertFrom field equal to TheDataFormat and a ConvertTo field equal to the Format field of one of the Facility Descriptors on TheFacilitiesList, such that the Facility Descriptor also has a Function field equal to TheFunction (step 150). For each match found, the matched parsing server capability descriptor is added to the list CapableParsingServersList (step 160). Then, the routine GetParsedData is executed (step 170). If, after GetParsedData is executed, the variable DataParsed? is TRUE (step 180), the routine ProcessReceivedData is executed (step 190).

If no match is found, or if the variable DataParsed? is FALSE after GetParsedData is executed, the routine FindCapableParsingServer is executed (step 200). Then the flag ParsingServerFound? is checked (step 210). If its value is FALSE, no appropriate parsing server was found, and the client system will be unable to perform the requested function on the specified resource. In

this case, an error message is returned to the user (step 220).

If ParsingServerFound? is TRUE, control returns to step 150.

Shown in FIG. 6 are the steps taken by the routine GetDataFormat. First, TheExtensionList is scanned to determine whether the TheURL contains an Extension matching the Extension field of one of the Extension Descriptors. If a match is found, the variable TheDataFormat is set to the Format of that Extension Descriptor (step 260).

If no match is found, a message is sent to the data server specified by TheURL, requesting it to transmit the Format_ID of the resource specified by TheURL (step 270). When the resulting Format_ID is received from the data server, the variable TheDataFormat is set accordingly (step 280).

Shown in FIG. 7 are the steps taken by the routine GetParsedData. First, the variable TheCurrentCapabilityDescriptor is set equal to the first parsing server capability descriptor on the CapableParsingServersList (step 300). Then, the variable TheParsingServer is set equal to the parsing server field of first parsing server capability descriptor on the list CapableParsingServersList (step 310), and the variable TheParsingServerOutputFormat is set to the ConvertTo field of that Descriptor (step 320). A PARSE_AND_TRANSMIT message is sent to TheParsingServer (step 330). Shown in FIG. 7a is a simplified diagram of a PARSE_AND_TRANSMIT message. The message contains a Label 340 indicating that it is a PARSE_AND_TRANSMIT message, a ConvertTo field 350 containing a Format_ID equal to the value of TheParsingServerOutputFormat, and a DataURL field 360 containing the contents of TheURL.

Except in cases of error conditions, a parsing

server will respond to a PARSE_AND_TRANSMIT message with a PARSE_AND_TRANSMIT_RESPONSE message. Shown in FIG. 7b is a simplified diagram of a PARSE_AND_TRANSMIT_RESPONSE message. The message contains a Label 370 indicating that it is a PARSE_AND_TRANSMIT_RESPONSE message, a DataURL field 380 containing the URL of the resource which was parsed, a Format field 390 containing a Format_ID, and a Data field 400, containing the parsed data. If, within a specified timeout period after the PARSE_AND_TRANSMIT message is sent, a non-error PARSE_AND_TRANSMIT_RESPONSE message is received from TheParsingServer (step 410), the variables TheData.Format and TheData.Data are set to the Format and Data fields of the PARSE_AND_TRANSMIT_RESPONSE message (step 420). The variable DataParsed? is set to TRUE (step 430), and the routine exits.

If a PARSE_AND_TRANSMIT_RESPONSE message is not received from TheParsingServer within a specified timeout period (or if an error message is received), the list CabableParsingServersList is checked to determine whether any additional Parser Serving capability descriptors remain on the list (step 440). If so, TheCurrentCapabilityDescriptor is set to the next unchecked parsing server capability descriptor on CapableParsingServersList (step 450). Control then returns to step 310. If not, the variable DataParsed? is set to FALSE (step 455), and the routine exits.

FIG. 8 shows the steps taken by the routine ProcessReceivedData. TheFacilitiesList is checked to determine whether there exists a Facility Descriptor with a Format equal to TheData.Format, and a Function equal to TheFunction (step 500). If so, the Facility identified by Facility Descriptor is used to process the data in TheData.Data (step 510). If not, an error is returned (step 520).

FIG. 9 shows the steps taken by the routine FindCapableParsingServer. First, the flag ParsingServerFound? is set to FALSE (step 600). Then, TheFacilitiesList is scanned to determine which Facility Descriptors have Function fields equal to TheFunction (step 610). For each descriptor found, a PARSING_SERVER_CAPABILITY_QUERY message is then broadcast over the WAN (step 620). Shown in FIG. 9a is a simplified diagram of the PARSING_SERVER_CAPABILITY_QUERY message. It contains a Label field identifying it as a PARSER_SERVER_CAPABILITY_QUERY message 630, a ConvertFrom field 640 and a ConvertTo field 650 containing Format_IDs. The ConvertFrom field will have the value of TheDataFormat, and the ConvertTo field will have the value of the Format field of the Facility Descriptor.

The client system then waits for a specified timeout period to receive PARSING_SERVER_CAPABILITY_RESPONSE messages (step 660). Shown in FIG. 9b is a simplified diagram of a PARSING_SERVER_CAPABILITY_RESPONSE message. It contains a Label 670 identifying it as a PARSING_SERVER_CAPABILITY_RESPONSE message, a MyID field 680 containing the Parsing_Server_ID of the parsing server, and a ConvertFrom field 690 and a ConvertTo field 700 both containing Format_IDs, indicating the parsing server's ability to convert data from format ConvertFrom to format ConvertTo.

For each PARSING_SERVER_CAPABILITY_RESPONSE message received, the routine UpdateParsingServerList is executed (step 710). FIG. 10 shows the steps taken by UpdateParsingServerList. First, the flag ParsingServerFound? is set to TRUE (step 800). Then a parsing server capability descriptor is added to the TheParsingServerList containing the ConvertFrom, ConvertTo and MyID fields of the

PARSING_SERVER_CAPABILTY_RESPONSE message (step 810).

FIG. 11 shows the steps taken by a parsing server in response to a PARSE_AND_TRANSMIT message. First, the Resource specified by the URL field of the PARSE_AND_TRANSMIT message is retrieved from the data server on which it is located (step 850). The data format of the retrieved data is then determined (step 860). TheParserList is then examined to determine whether there is a Parser capability descriptor having a ConvertFrom field equal to the format of the data, and a ConvertTo field equal to the ConvertTo field of the PARSE_AND_TRANSMIT message (step 870). If a matching Parser capability descriptor is found, the data is translated to the ConvertTo format using the Parser specified by the Parser capability descriptor (step 880). The resulting translated data is then transmitted back to the client system (step 890). If no matching Parser Capability Descriptor is found, an error message is transmitted back to the client system (step 900).

FIG. 12 shows the steps taken by a parsing server in response to a PARSING_SERVER_CAPABILITY_QUERY. First, a pointer is set to the first Parser capability descriptor on TheParserList (step 1000). If the ConvertFrom and ConvertTo fields of the PARSING_SERVER_CAPABILITY_QUERY message equal the ConvertFrom and ConvertTo fields of the Parser capability descriptor (step 1010), a PARSER_SERVER_CAPABILITY_RESPONSE message is then generated containing the ConvertFrom and ConvertTo fields of the Parser capability descriptor, as well as the Parsing_Server_ID of the parsing server (step 1020). If the ConvertFrom and ConvertTo fields do not match, TheParserList is checked to see whether there remain addition Parser capability descriptors (step 1030). If so, then the pointer is then set to the next such

Descriptor (step 1040) and control returns to step 1010.

The invention may be implemented in digital electronic circuitry or in computer hardware, firmware, software, or in combinations of them. Apparatus of the invention may be implemented in a computer program product tangibly embodied in a machine readable storage device for execution by a computer processor; and method steps of the invention may be performed by one or more computer processors executing a program to perform functions of the invention by operating on input data and generating output. Suitable processors include, by way of example, both general and special purpose microprocessors. Generally, a processor will receive instructions and data from a read-only memory and/or a random access memory. Storage devices suitable for tangibly embodying computer program instructions include all forms of non-volatile memory, including by way of example semiconductor memory devices, such as EPROM, EEPROM, and flash memory devices; magnetic disks such as internal hard disks and removable disks; magneto-optical disks; and CD-ROM disks. Any of the foregoing may be supplemented by, or incorporated in, specially-designed ASICs (application-specific integrated circuits).

Other embodiments are within the scope of the following claims. For example, the order of performing steps of the invention or the ordering, structuring or naming of data structures of the invention may be changed by those skilled in the art and still achieve desirable results.

A client system may rely, for example, on only one specified parsing server to convert data.

Parsing servers, at intervals, may broadcast messages over the WAN announcing their capabilities, and client systems may update their parsing server lists

accordingly.

Client systems, at intervals, may broadcast messages over the WAN asking parsing servers to identify themselves and their capabilities and the client systems may use the responses to update their parsing server lists accordingly.

A client system may keep track of errors encountered when attempting to utilize a given parsing server, and delete the corresponding parsing server capability descriptor from TheParsingServerList if errors occurred at more than a specified frequency.

A client system may initially obtain the file specified by TheURL from the File Server and determine its Format. If it cannot process the file with its own facilities, it may engage in a process of finding appropriate parsing servers such that each PARSE_AND_TRANSMIT message includes the actual file instead of the URL for the file, and the parsing server parses the included file, rather than obtaining it from a File Server.

TheParsingServerList may include a list of parsing servers, without capability information, and when a client system needs to convert a given file, it may query all of the parsing servers on TheParsingServerList to determine which, if any, is capable of doing the required transformation.

Client systems may, alternatively, not maintain a parsing server list. If a parsing server is needed, the client system may broadcast a query message over the WAN to determine whether any parsing servers could perform the needed conversion. The client system may use a parsing server which responds affirmatively to its query message.

A client system may request one parsing server to translate data to an intermediate format and then after

receiving the translated data, request a second parsing server to translate the data from the intermediate format to a final format.

The TheParsingServer may be chosen from the CapableParsingServerList according to a prioritization heuristic.

What is claimed is:

1. In a data network in which a multiplicity of data formats exist, a method for performing a particular function on data represented in a first data format, on a client system lacking resources to perform the function on data in the first data format but having resources to perform the function on data in a second data format, the method comprising:

transmitting a request message from the client system to a parsing server;

transmitting the data from a data server to the parsing server; parsing the data at the parsing server into the second data format;

receiving on the client system the data parsed into the second format from the parsing server; and

using a resource on the client system to perform the function on the parsed data.

2. The method of claim 1, wherein the request message comprises a Uniform Resource Locator specifying the location of the data on the network.

3. The method of claim 1, wherein the client system is the same as the data server.

4. The method of claim 1, wherein the request message specifies the second data format.

5. The method of claim 1, wherein the request message specifies the first data format.

6. The method of claim 1, wherein the parsing server is selected from a pre-existing list of parsing servers.

7. The method of claim 1, wherein the parsing server is selected from among a set of parsing servers which respond to a request broadcast to the network.

8. The method of claim 6, wherein the pre-existing list of parsing servers is generated using information broadcast to the network by parsing servers.

9. The method of claim 6, wherein the pre-existing list of parsing servers is generated using information

sent by parsing servers in response to messages broadcast to the network by the client system.

10. The method of claim 6, wherein the step of selecting the parsing server comprises:

identifying the first data format;

identifying which resources on the client can perform the function;

identifying a set of data formats upon which the resources can perform the function;

identifying the translation capabilities of parsing servers on the list;

identifying a set of parsing servers which are capable of translating data from the first data format to one of the set of data formats; and

choosing a parsing server from among the set of parsing servers which is capable of translating from the first data format to a data format from the set of data formats.

11. The method of claim 10, wherein the step of identifying the translation capabilities of parsing servers comprises accessing data stored on the client system.

12. The method of claim 10, wherein the step of identifying the translation capabilities of parsing servers comprises sending messages to parsing servers over the network requesting them to indicate their translation capabilities and receiving messages from parsing servers indicating their capabilities.

13. The method of claim 1, wherein the network is a wide area network.

14. The method of claim 1, wherein the network is a local area network.

15. A computer program, residing on a computer-readable medium, comprising instructions for causing a computer connected to a network to:

receive a request message from a client system over the network;

receive data from a data server in a first format over the network;

parse the data into a second data format; and

transmit the data in the second format over the network to the client system.

16. A parsing server for use on a network, comprising a digital computer and a computer program storage medium having a computer program stored thereon for execution on the digital computer, the program comprising instructions for causing the computer to:

receive a request message from a client system over the network;

receive data from a data server in a first format over the network;

parse the data into a second data format;

transmit the data in the second format over the network to the client system.

17. In a network in which a multiplicity of data formats are recognized, a method for performing a particular function on data represented in a first data format, on a client system lacking resources to perform the function on data in the first data format, but having resources to perform the function on data in at least one other data format, the method comprising:

identifying the first data format;

identifying which resources on the client can perform the function;

identifying a set of data formats upon which the resources can perform the function;

identifying the translation capabilities of parsing servers on a list of parsing servers;

identifying the set of parsing servers which are capable of translating data from the first data format to

one of the set of data formats;

choosing a parsing server from among the set of parsing servers;

sending a request message from the client system to the chosen parsing server, specifying the URL of the data, the first data format, and the second data format;

sending the data specified by the URL from a data server to the parsing server;

on the parsing server parsing the data into the second data format;

sending the parsed data in the second data format from the parsing server to the client system;

using a resource on the client system to perform the function on the parsed data in the second data format.

18. In a data network including server systems and client systems, a method of converting data from a first format unusable by a client system into a second format usable by the client system comprising the steps of:

(a) directing data in the first format at the request of the client system to a parsing server capable of converting such data from the first format to the second format;

(b) parsing the data at the parsing server to the second format;

(c) directing the data in the second format from the parsing server to the client system; and

(d) using the data in the second format in the client system.

19. In a data network including server systems and client systems, a method of converting data from a first format unusable by a client system into a second format usable by the client system comprising the steps of sending the data to a parsing server capable of converting such data from the first format to the second

format, parsing the data in the parsing server from the first format to the second format, and sending the data in the second format to the client system.

FIG. 1a

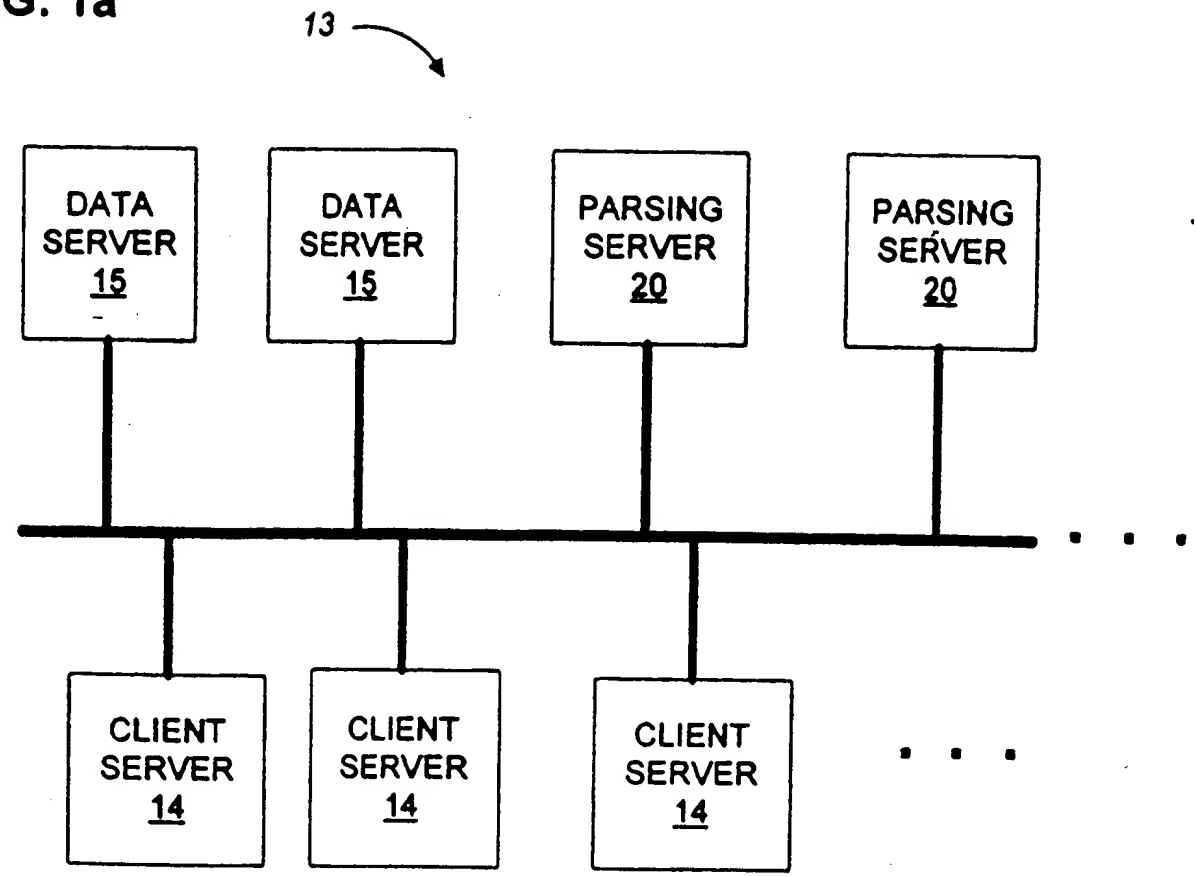

13
DATA SERVER 15
DATA SERVER 15
PARSING SERVER 20
PARSING SERVER 20
CLIENT SERVER 14
CLIENT SERVER 14
CLIENT SERVER 14

FIG. 1b

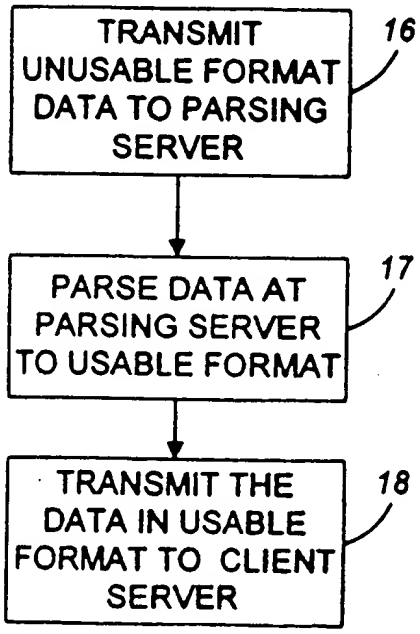

TRANSMIT UNUSABLE FORMAT DATA TO PARSING SERVER
16
PARSE DATA AT PARSING SERVER TO USABLE FORMAT
17
TRANSMIT THE DATA IN USABLE FORMAT TO CLIENT SERVER
18

FIG. 2

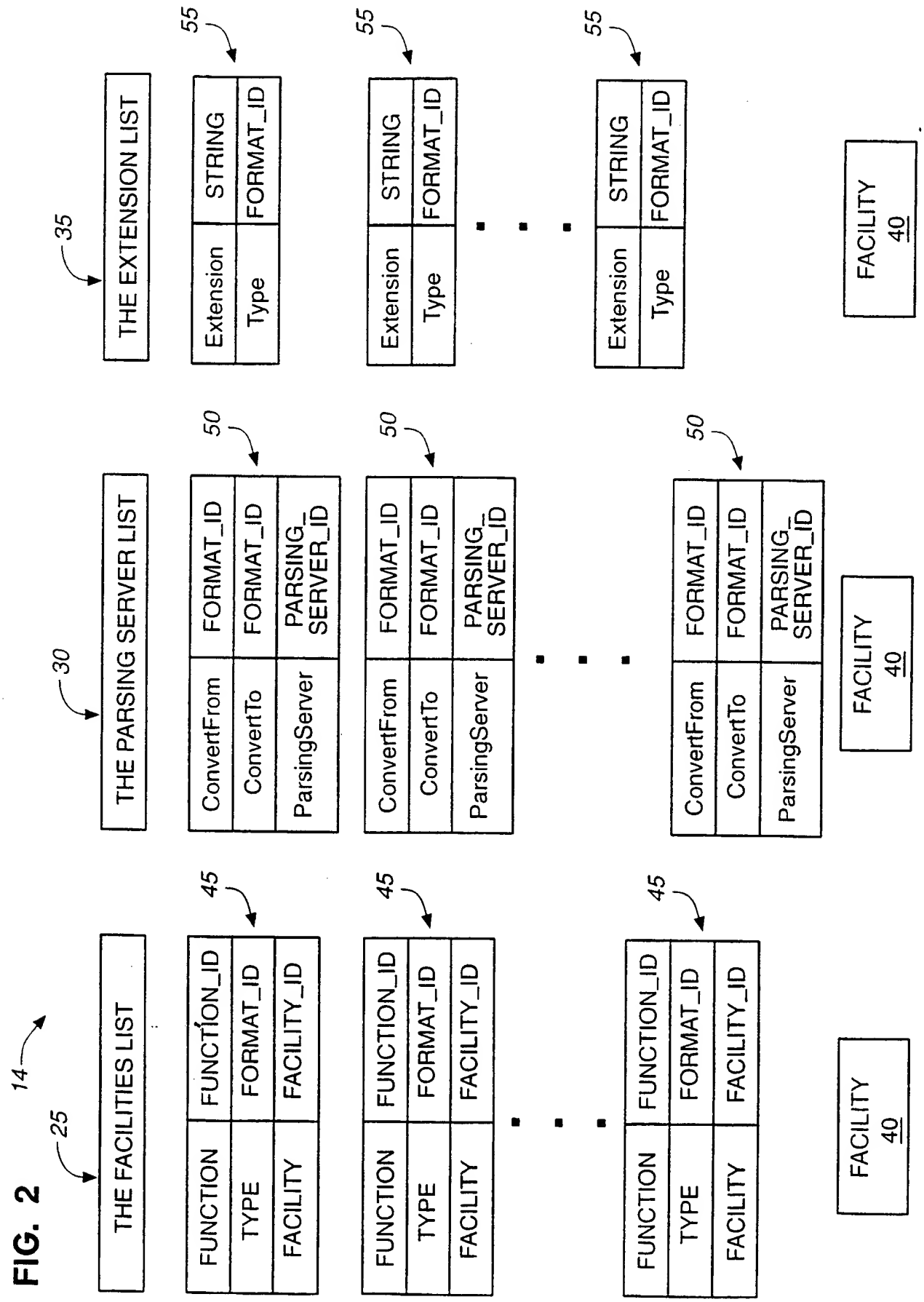

14
25
THE FACILITIES LIST
FUNCTION
TYPE
FACILITY
45
FUNCTION_ID
FORMAT_ID
FACILITY_ID
30
THE PARSING SERVER LIST
50
ConvertFrom
ConvertTo
ParsingServer
FORMAT_ID
FORMAT_ID
PARSING_SERVER_ID
35
THE EXTENSION LIST
55
Extension
Type
STRING
FORMAT_ID
FACILITY 40
FACILITY 40
FACILITY 40

FIG. 3

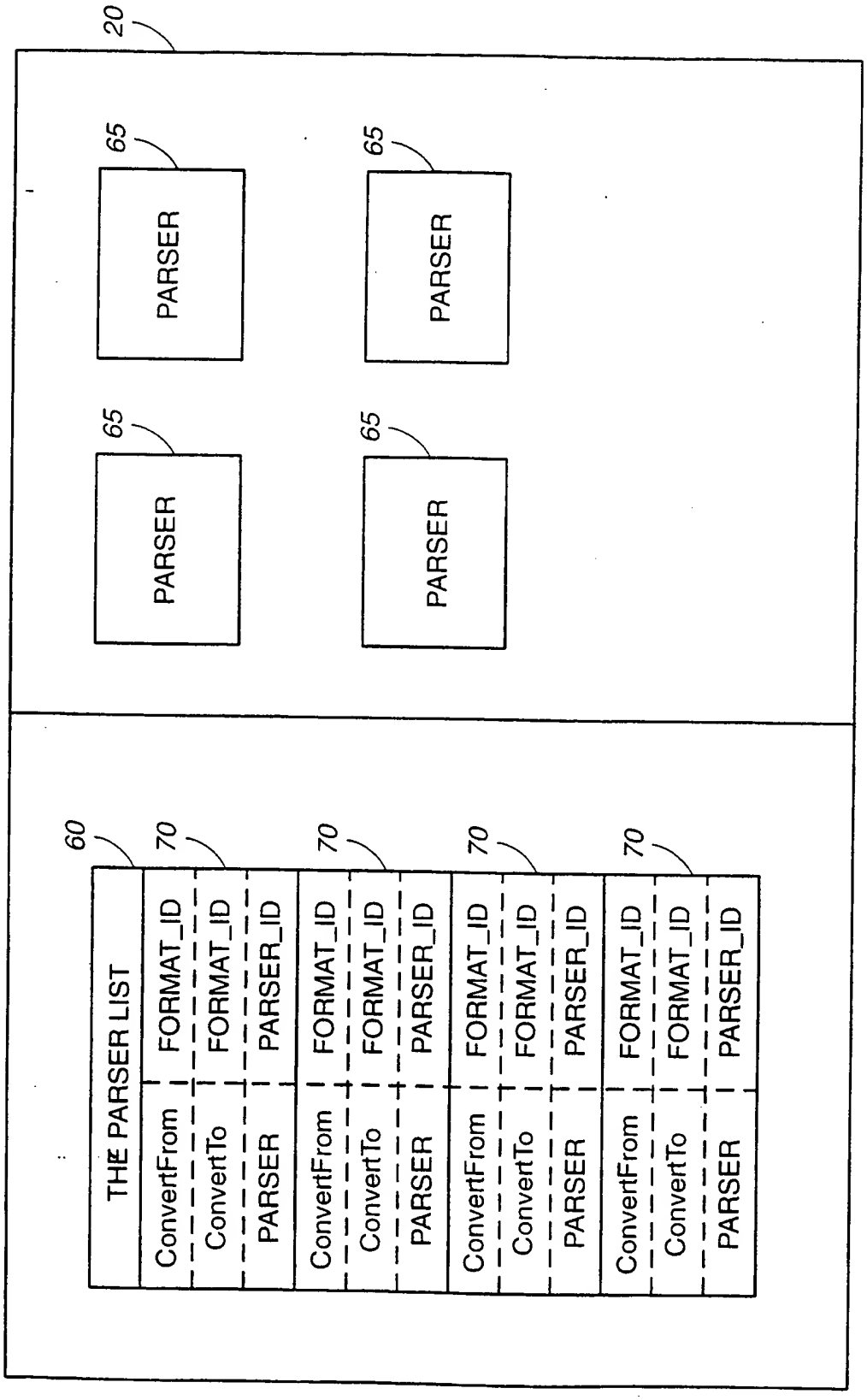

20
65
PARSER
65
PARSER
65
PARSER
65
PARSER
60
THE PARSER LIST
70
FORMAT_ID
ConvertFrom
FORMAT_ID
ConvertTo
PARSER_ID
PARSER
FORMAT_ID
ConvertFrom
FORMAT_ID
ConvertTo
PARSER_ID
PARSER
FORMAT_ID
ConvertFrom
FORMAT_ID
ConvertTo
PARSER_ID
PARSER
FORMAT_ID
ConvertFrom
FORMAT_ID
ConvertTo
PARSER_ID
PARSER

FIG. 4

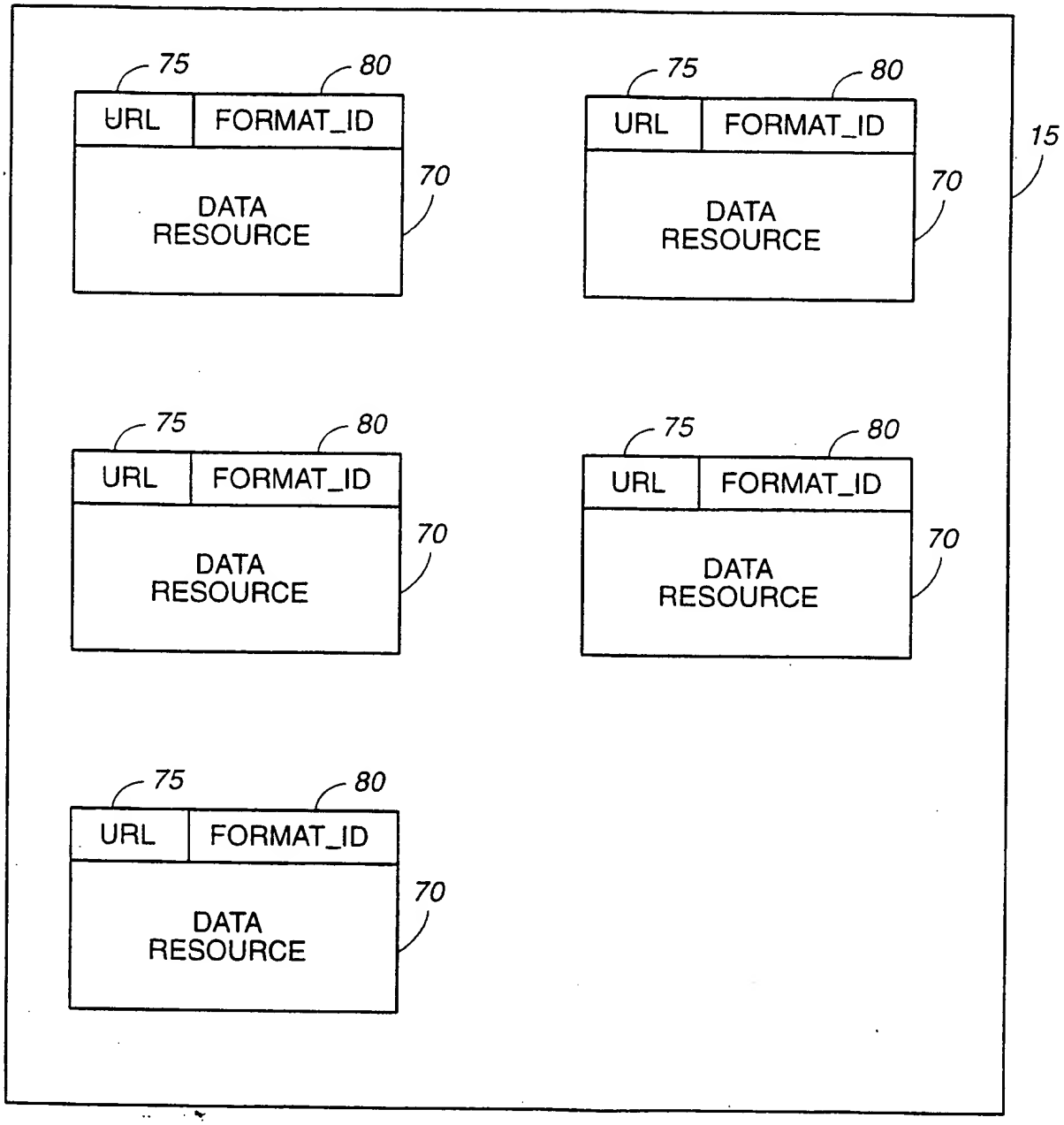


FIG. 5

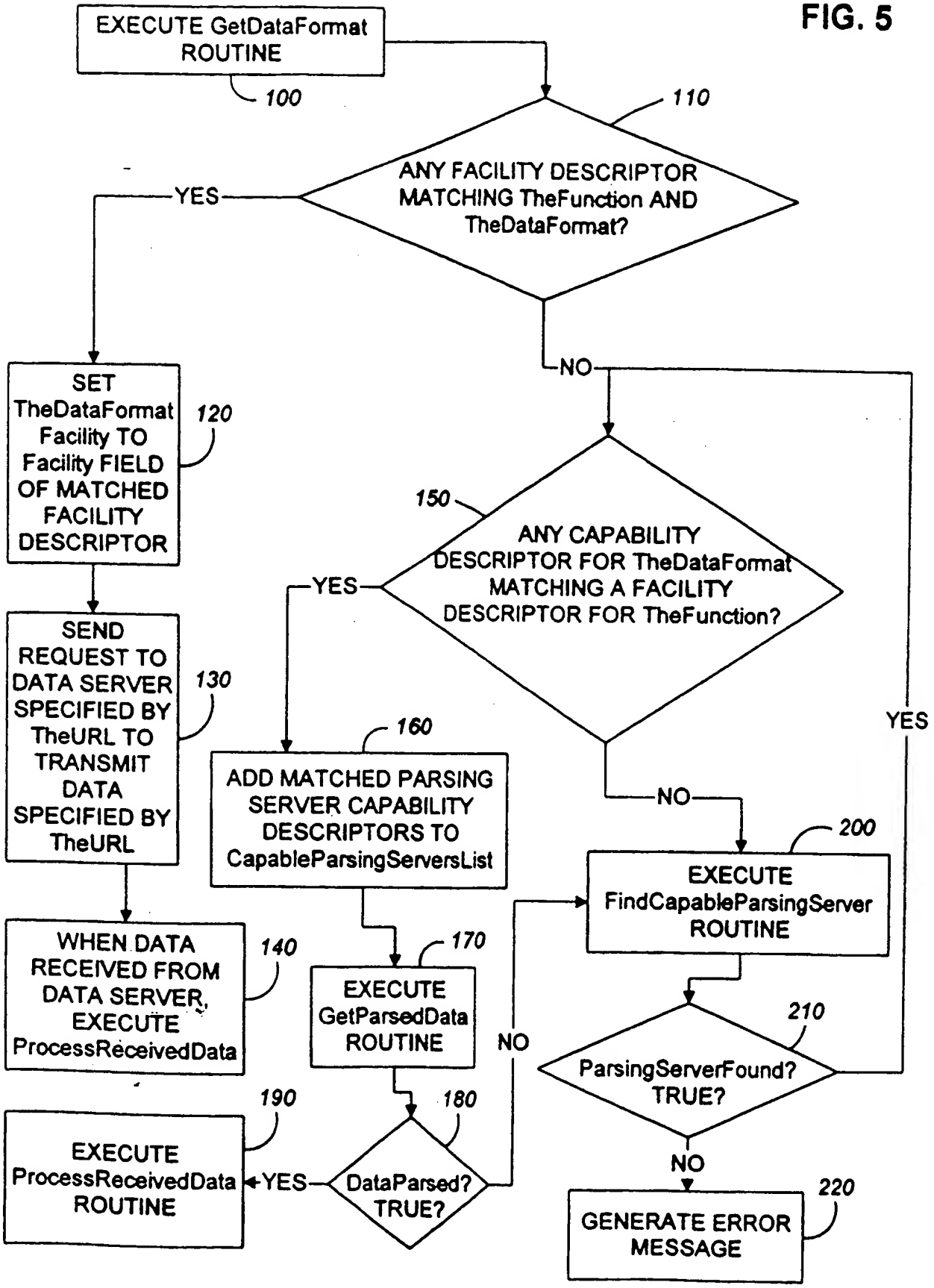

EXECUTE GetDataFormat ROUTINE
100
110
ANY FACILITY DESCRIPTOR MATCHING TheFunction AND TheDataFormat?
YES
NO
SET TheDataFormat Facility TO Facility FIELD OF MATCHED FACILITY DESCRIPTOR
120
SEND REQUEST TO DATA SERVER SPECIFIED BY TheURL TO TRANSMIT DATA SPECIFIED BY TheURL
130
WHEN DATA RECEIVED FROM DATA SERVER, EXECUTE ProcessReceivedData
140
150
ANY CAPABILITY DESCRIPTOR FOR TheDataFormat MATCHING A FACILITY DESCRIPTOR FOR TheFunction?
YES
NO
160
ADD MATCHED PARSING SERVER CAPABILITY DESCRIPTORS TO CapableParsingServersList
170
EXECUTE GetParsedData ROUTINE
180
DataParsed? TRUE?
YES
NO
190
EXECUTE ProcessReceivedData ROUTINE
200
EXECUTE FindCapableParsingServer ROUTINE
210
ParsingServerFound? TRUE?
YES
NO
220
GENERATE ERROR MESSAGE

FIG. 6

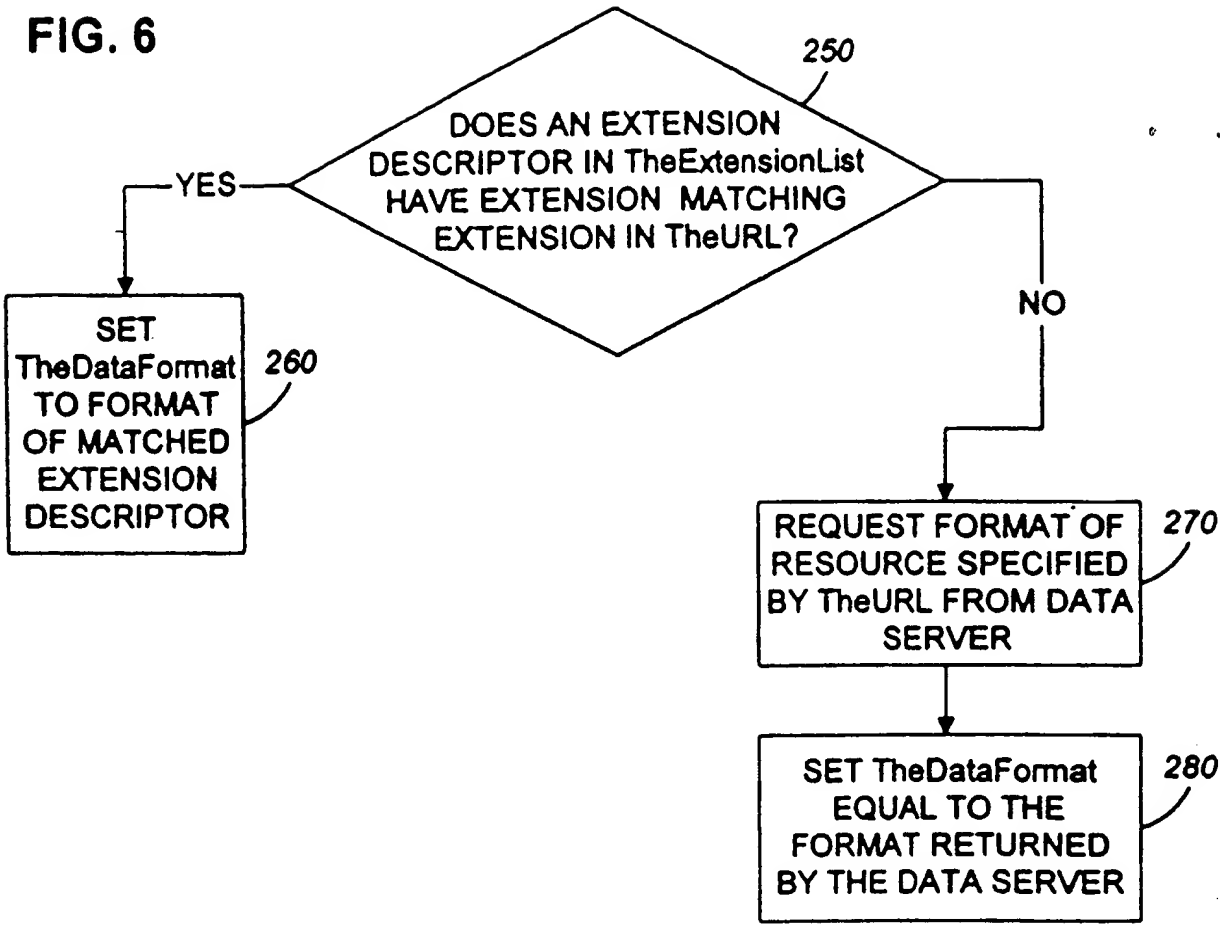

250
DOES AN EXTENSION DESCRIPTOR IN TheExtensionList HAVE EXTENSION MATCHING EXTENSION IN TheURL?
YES
NO
SET TheDataFormat TO FORMAT OF MATCHED EXTENSION DESCRIPTOR
260
REQUEST FORMAT OF RESOURCE SPECIFIED BY TheURL FROM DATA SERVER
270
SET TheDataFormat EQUAL TO THE FORMAT RETURNED BY THE DATA SERVER
280

FIG. 8

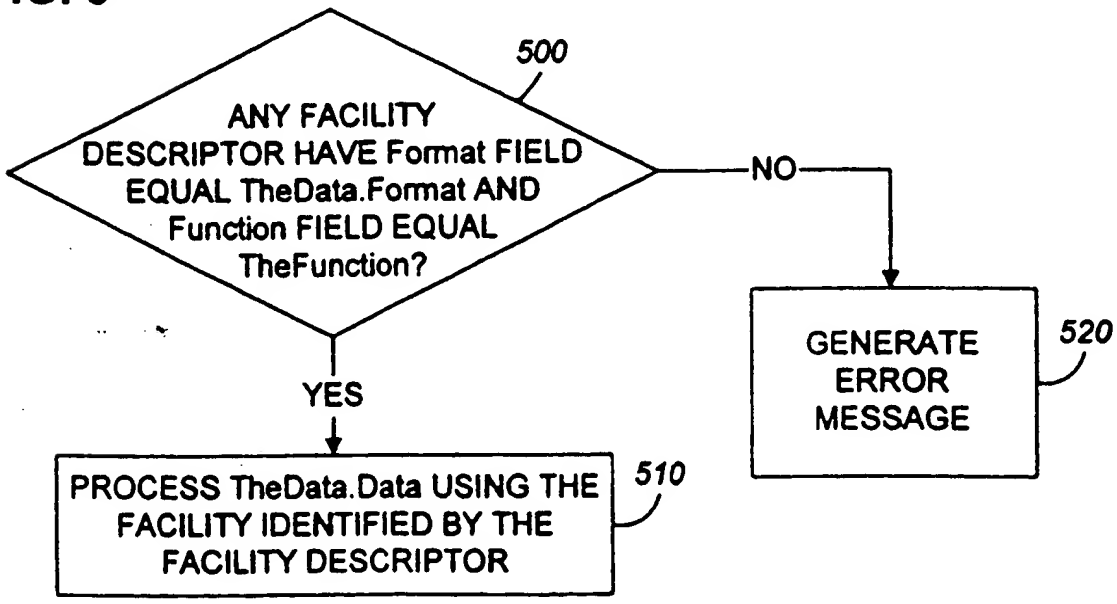

500
ANY FACILITY DESCRIPTOR HAVE Format FIELD EQUAL TheData.Format AND Function FIELD EQUAL TheFunction?
NO
YES
GENERATE ERROR MESSAGE
520
PROCESS TheData.Data USING THE FACILITY IDENTIFIED BY THE FACILITY DESCRIPTOR
510

FIG. 7

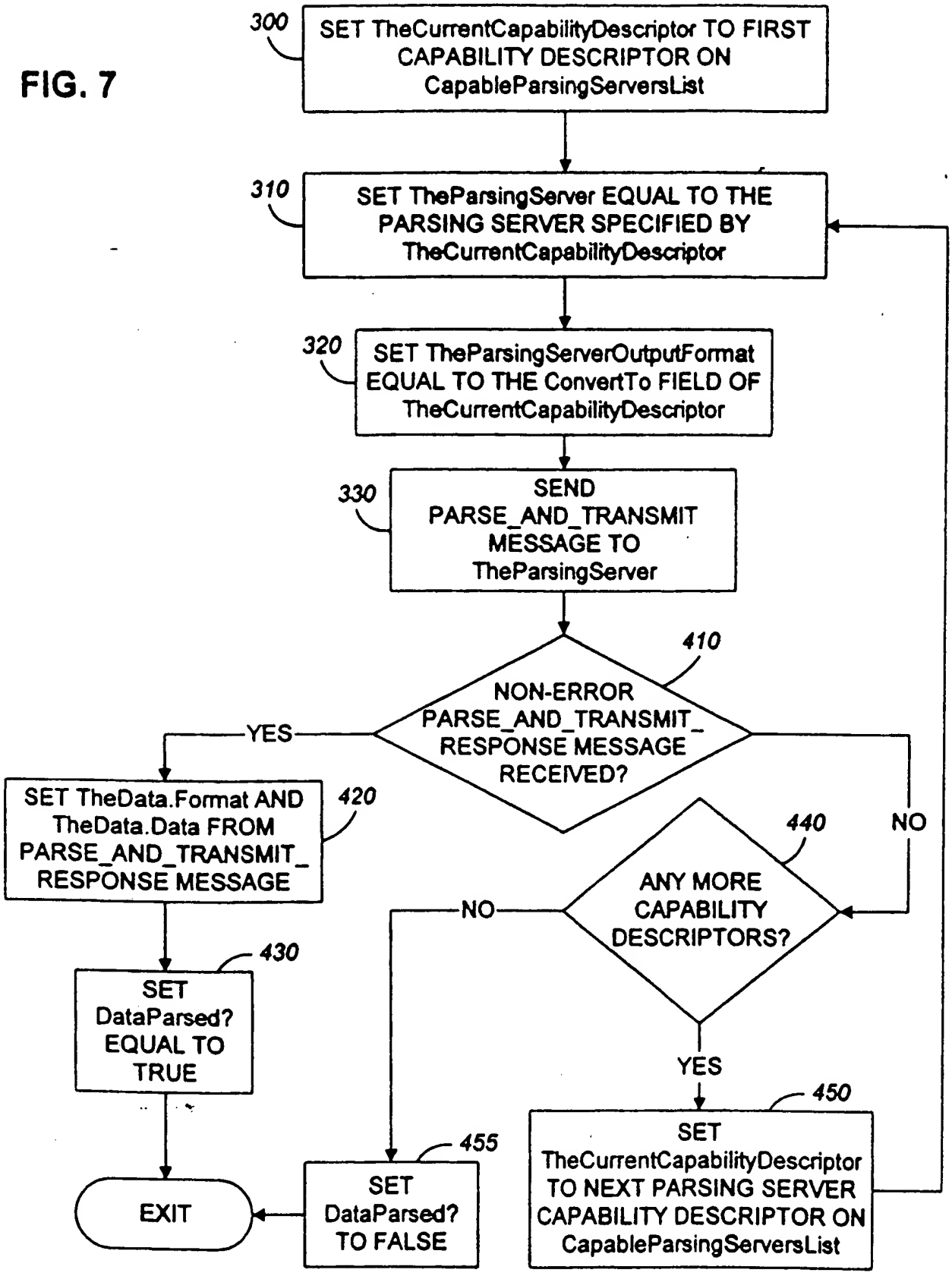

300
SET TheCurrentCapabilityDescriptor TO FIRST CAPABILITY DESCRIPTOR ON CapableParsingServersList
310
SET TheParsingServer EQUAL TO THE PARSING SERVER SPECIFIED BY TheCurrentCapabilityDescriptor
320
SET TheParsingServerOutputFormat EQUAL TO THE ConvertTo FIELD OF TheCurrentCapabilityDescriptor
330
SEND PARSE_AND_TRANSMIT MESSAGE TO TheParsingServer
410
NON-ERROR PARSE_AND_TRANSMIT_ RESPONSE MESSAGE RECEIVED?
YES
NO
420
SET TheData.Format AND TheData.Data FROM PARSE_AND_TRANSMIT_ RESPONSE MESSAGE
440
ANY MORE CAPABILITY DESCRIPTORS?
NO
YES
430
SET DataParsed? EQUAL TO TRUE
450
SET TheCurrentCapabilityDescriptor TO NEXT PARSING SERVER CAPABILITY DESCRIPTOR ON CapableParsingServersList
455
SET DataParsed? TO FALSE
EXIT

FIG. 7a

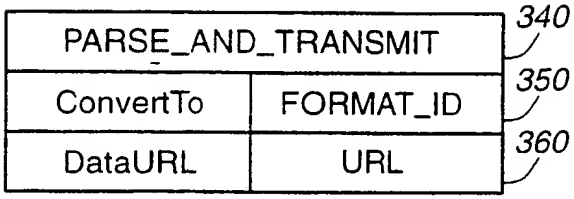


FIG. 7b

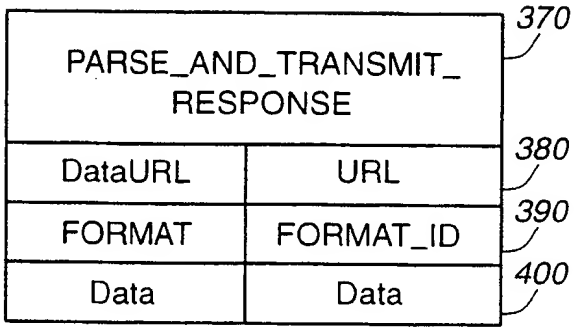


FIG. 9a

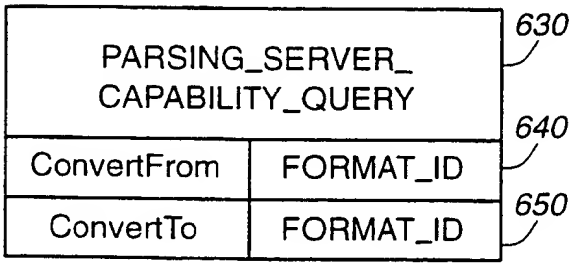


FIG. 9b

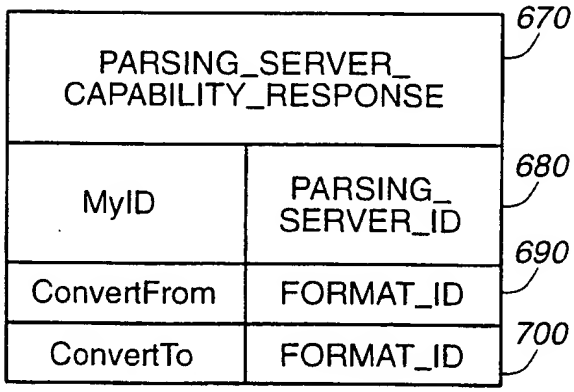


FIG. 9

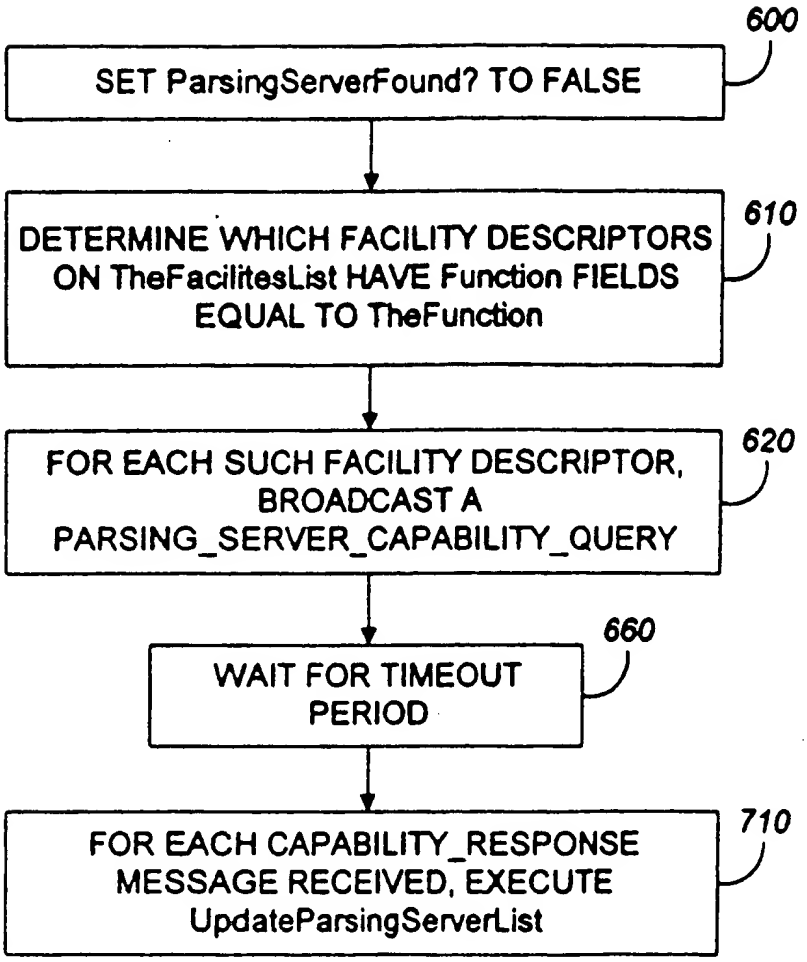

600
SET ParsingServerFound? TO FALSE
610
DETERMINE WHICH FACILITY DESCRIPTORS ON TheFacilitesList HAVE Function FIELDS EQUAL TO TheFunction
620
FOR EACH SUCH FACILITY DESCRIPTOR, BROADCAST A PARSING_SERVER_CAPABILITY_QUERY
660
WAIT FOR TIMEOUT PERIOD
710
FOR EACH CAPABILITY_RESPONSE MESSAGE RECEIVED, EXECUTE UpdateParsingServerList

FIG. 10

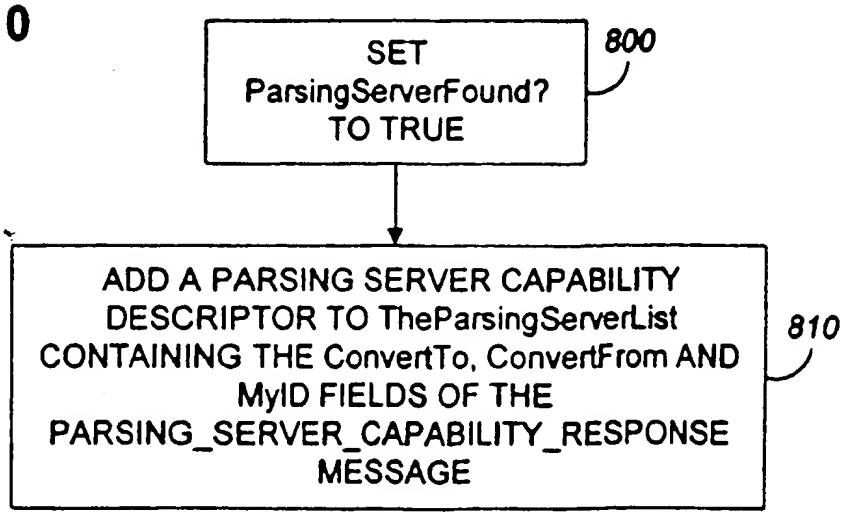

800
SET ParsingServerFound? TO TRUE
810
ADD A PARSING SERVER CAPABILITY DESCRIPTOR TO TheParsingServerList CONTAINING THE ConvertTo, ConvertFrom AND MyID FIELDS OF THE PARSING_SERVER_CAPABILITY_RESPONSE MESSAGE

FIG. 11

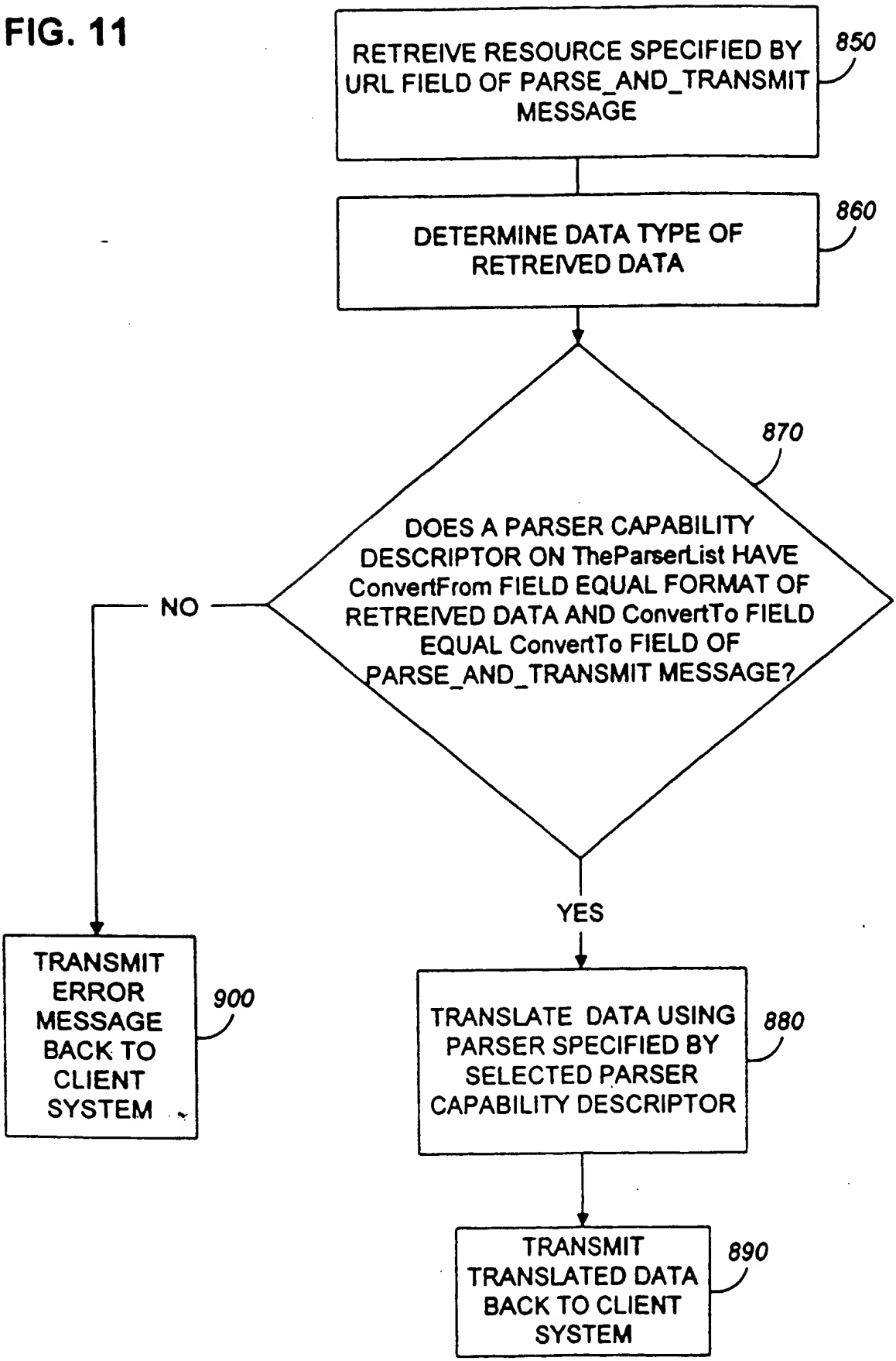
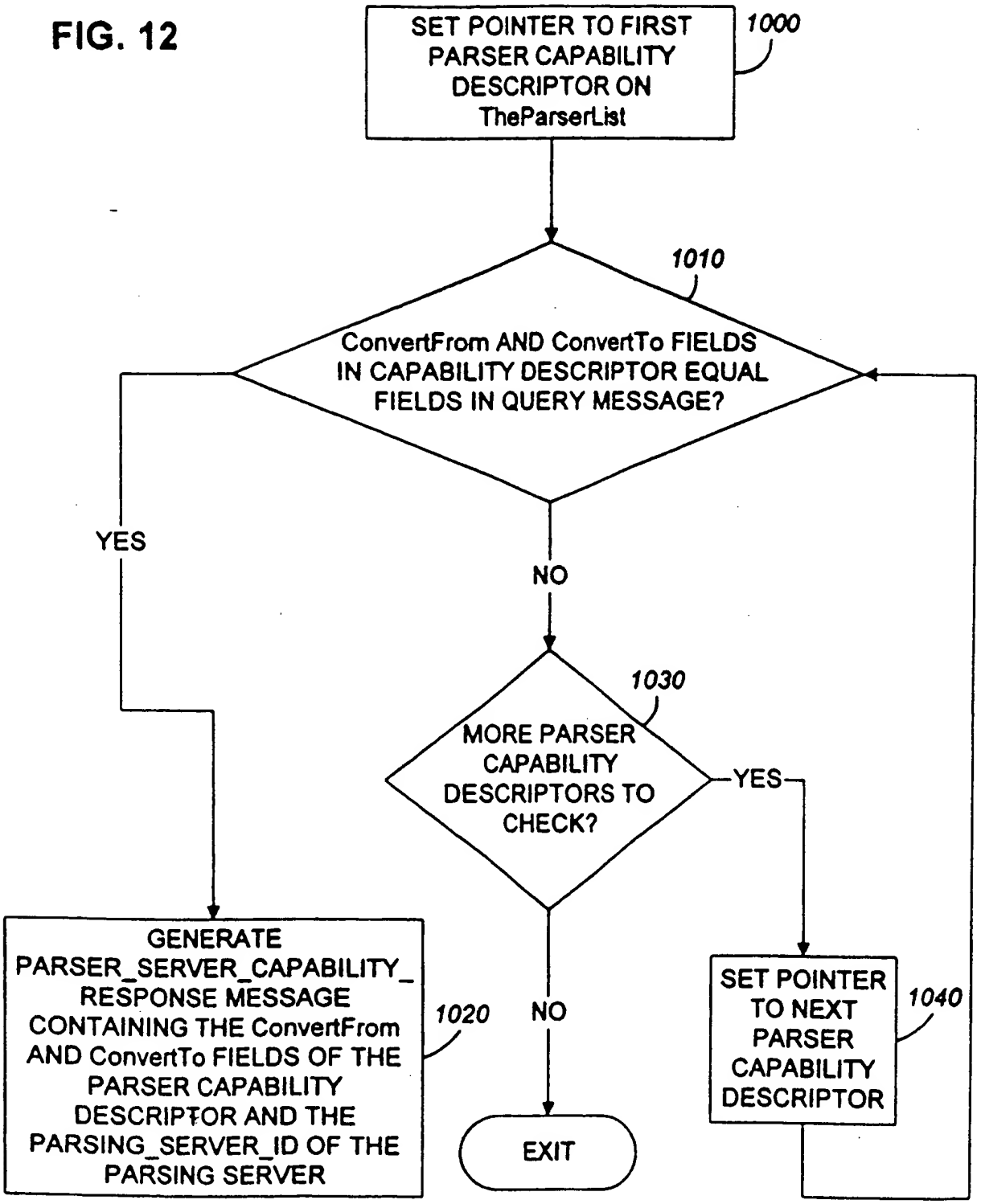

RETREIVE RESOURCE SPECIFIED BY URL FIELD OF PARSE_AND_TRANSMIT MESSAGE
850
DETERMINE DATA TYPE OF RETREIVED DATA
860
870
DOES A PARSER CAPABILITY DESCRIPTOR ON TheParserList HAVE ConvertFrom FIELD EQUAL FORMAT OF RETREIVED DATA AND ConvertTo FIELD EQUAL ConvertTo FIELD OF PARSE_AND_TRANSMIT MESSAGE?
NO
YES
TRANSMIT ERROR MESSAGE BACK TO CLIENT SYSTEM
900
TRANSLATE DATA USING PARSER SPECIFIED BY SELECTED PARSER CAPABILITY DESCRIPTOR
880
TRANSMIT TRANSLATED DATA BACK TO CLIENT SYSTEM
890

FIG. 12


SET POINTER TO FIRST PARSER CAPABILITY DESCRIPTOR ON TheParserList
1000
1010
ConvertFrom AND ConvertTo FIELDS IN CAPABILITY DESCRIPTOR EQUAL FIELDS IN QUERY MESSAGE?
YES
NO
1030
MORE PARSER CAPABILITY DESCRIPTORS TO CHECK?
YES
NO
GENERATE PARSER_SERVER_CAPABILITY_ RESPONSE MESSAGE CONTAINING THE ConvertFrom AND ConvertTo FIELDS OF THE PARSER CAPABILITY DESCRIPTOR AND THE PARSING_SERVER_ID OF THE PARSING SERVER
1020
SET POINTER TO NEXT PARSER CAPABILITY DESCRIPTOR
1040
EXIT

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US98/09033

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) : G06F 5/01; H04B 1/00
US CL : 395/200.33, 200.47, 200.48, 200.49, 200.61, 200.76; 340/825.52
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 395/200.33, 200.47, 200.48, 200.49, 200.61, 200.76; 340/825.52

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5,727,159 [KIKINIS] 10 Mar 1998, fig 4, col 9, line 61 to col 10, line 35; col 2, lines 54-61; col 3, lines 18-30	1-5, 13-16, 18-19
---	---	---
Y	col 8, lines 49-62	6-12, 17
Y	US 5,227,778 [VACON et al] 13 Jul 1993, col 2, lines 17-34; col 2, lines 1-11; col 2, lines 34-45	6-12, 17

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search	Date of mailing of the international search report
10 SEPTEMBER 1998	22 OCTOBER 1998
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	Authorized officer QUOC-KHANH LE